Induction of Minimalist Grammars over Morphemes

Marina Ermolaeva
University of Chicago
mermolaeva@uchicago.edu

1 Introduction

Syntactic literature tends towards a big-picture outlook, abstracting away from details such as full specifications of lexical items or features involved in derivations. However, a lower-level description is required to identify differences between competing analyses of the same phenomenon.

For a concrete example, consider the double object construction (e.g. *John gave Mary a book*) in English. One option is to combine the internal arguments *Mary* and *a book* in a "small clause" or PP-like structure and then merge the verb with this constituent (e.g. Kayne 1984; Pesetsky 1996; Harley and Jung 2015). The alternative is to have the verb select the arguments one by one, giving rise to VP-shells (Larson, 1988) and analyses inspired by them (Kawakami, 2018).

It is natural to ask whether it would be possible, assuming a sufficiently rich formalism compatible with the Minimalist framework, to choose the answer to this and similar questions based on some robust quantitative metric.

2 Minimalist grammars

Minimalist grammars (Stabler, 1997) are a natural choice for this task. As a formalization of Chomsky's (1995) Minimalist Program, they are well-suited for implementing analyses of syntactic phenomena, yet at the same time explicit regarding the assumptions about syntactic units and operations.

Minimalist grammars define lexical items (atomic expressions) as pairs consisting of a phonetic exponent and a sequence of syntactic features (1). The first feature of each lexical item is accessible to the operations, Merge and Move, that target and delete matching features of opposing polarities. Merge combines two expressions to build a new one, whereas Move is unary and attracts a sub-expression into the specifier of the main structure. Merge with head movement (HM) concatenates pronounced features of the heads of its arguments, providing a simple implementation of concatenative morphology.

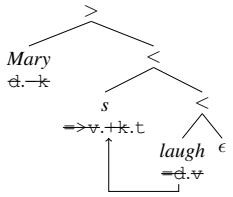
(1)

	Positive polarity	Negative polarity
Merge	=x (right selector) =>x (HM selector) x= (left selector)	x (category)
Move	+x (licensor)	−x (licensee)

Whichever expression contributed the positive feature becomes the head of the new expression. A complete sentence is an expression with no features left but the category t on its head. An example lexicon is given in (2), along with the derived tree of the sentence *Mary laughs* generated by it.

(2)

Mary :: d.−k
s :: =>v.+k.t
ed :: =>v.+k.t
laugh :: =d.v
jump :: =d.v



3 Learning from dependencies

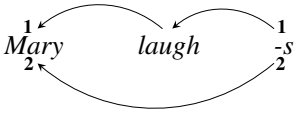
There is a substantial body of work dedicated to learning grammars from unstructured strings; e.g. an overview in (Clark, 2017). In particular, Yoshinaka (2011) presents an algorithm for learning certain subclasses of multiple context-free grammars. One can construct an equivalent Minimalist grammar for any multiple context-free grammar (Michaelis, 2001). However, such a grammar would not make for a good starting point if our goal is to compare and evaluate proposals of theoretical syntax, as modern syntactic theory heavily relies on highly abstract concepts such as empty categories, not directly visible in the raw data.

On the other hand, Siskind (1996) suggests that rather than obtain syntactic structure from unstructured input, the learner can start the process of

Proceedings of the Society for Computation in Linguistics (SCiL) 2020, pages 484-487.
New Orleans, Louisiana, January 2-5, 2020

grounding, or mapping linguistic units to atoms of meaning, before learning syntax. Then it is plausible that the learner can identify relations formed by Merge and Move before knowing what lexical items or syntactic features are involved, which gives rise to the approach to learning proposed by Kobele et al. (2002). For each sentence the learner is given ordered and directed dependencies between morphemes, with suffixes marked as such (3).

(3)



In this scenario, full lexical items (unique for each sentence) can be recovered from the dependencies. The learner's task is to determine which feature distinctions should be kept and which need to be collapsed, or unified. The pressure for unification comes from a restriction on the number of homophonous lexical items (Kanazawa, 1995).

As an illustration, consider the corpus of two sentences, *Mary laugh -s* and *Mary laugh -ed*. The learner assembles lexical items by assigning a fresh feature to each dependency, assuming that each data point is a complete sentence of category `t`. The ordering of dependencies determines whether each of them corresponds to Merge or Move. The initial lexicon (4) contains two copies each of *Mary* and *laugh*.

(4)

Mary :: `f1.-f2`	*Mary* :: `f4.-f5`
laugh :: `=f1.f3`	*laugh* :: `=f4.f6`
-s :: `=>f3.+f2.t`	*-ed* :: `=>f6.+f5.t`

The final step is to rename the corresponding features throughout the lexicon in order to collapse each pair of items into one. A familiar-looking lexicon will arise if `f1` and `f4` are mapped to `d`, `f2` and `f5` to `k`, and `f3` and `f6` to `v`. After feature unification, the grammar shrinks from six to four lexical items, which can still derive the input sentences.

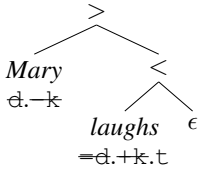
4 Lexical item decomposition

This paper builds on (Kobele et al., 2002), aiming to relax the segmentation requirement and let the algorithm learn the structure within complex words and any generalizations it would lead to.

Compare the lexicon in (2) with (5), which generates exactly the same set of sentences. Intuitively, (2) is better than (5), even though both have the same number of lexical items. It captures the similarities between different forms of the same verb and recognizes the verbs' internal structure: two correct generalizations that (5) misses.

(5)

Mary :: `d.-k`
laughs :: `=d.+k.t`
laughed :: `=d.+k.t`
jumps :: `=d.+k.t`
jumped :: `=d.+k.t`



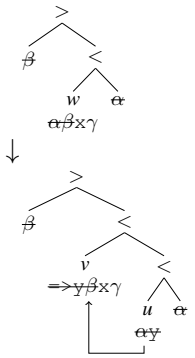
This difference can be quantified in a number of ways – naively as the number of phonetic and/or syntactic units, length of encoding the grammar or, taking into account the cost of encoding the corpus, as minimum description length (Rissanen, 1978).

How to transition from a grammar over words such as (5) to a grammar over morphemes (2)? In linguistic terms, the latter reanalyzes the verb as a complex head formed by head movement. This can be generalized to a *decomposition* operation (Kobele, 2018) that splits a lexical item's syntactic and phonetic features, producing a new item with a fresh category (6). The morphological operation generating w from the stem u and suffix v is denoted by $\oplus$; in the simplest case it corresponds to string concatenation.

(6)

w :: $\alpha\beta\mathtt{x}\gamma$

u :: $\alpha\mathtt{y}$
v :: $=\!>\mathtt{y}\beta\mathtt{x}\gamma$
$w = u \oplus v$



If syntactic decomposition is not accompanied by splitting the phonological material, one of the new lexical items will be an empty functional head. Otherwise, the algorithm has to construct a morphological rule by searching for phonological similarities across the lexicon.

Concatenative morphology has been shown to be successfully learnable in an unsupervised scenario (Goldsmith, 2001), with a possibility of using the results to infer the syntactic category of words (Hu et al., 2005); the problem of irregular and non-concatenative patterns (such as *sings* vs. *sang*) is also addressed in the literature (e.g. Lee

and Goldsmith 2014). Thus, in our case the learner has access to two separate sources of information – syntactic features and phonological patterns – to base its decisions on.

Multiple lexical items sharing a sub-sequence of syntactic features can be decomposed simultaneously, factoring out the shared features. The pressure to do this comes from a reduced cost in features; replacing repeating sequences is a well-known compression technique (cf. Nevill-Manning et al. 1994).

5 Towards a grammar over morphemes

The following example shows how a naive word-based grammar can be transformed into a linguistically motivated grammar over morphemes via decomposition and feature unification. Let the learner start with dependency structures (over non-segmented words) for the following eight sentences:

(7)

Mary laughs	*Mary jumps*
Mary is laughing	*Mary is jumping*
Mary will laugh	*Mary will jump*
Mary will be laughing	*Mary will be jumping*

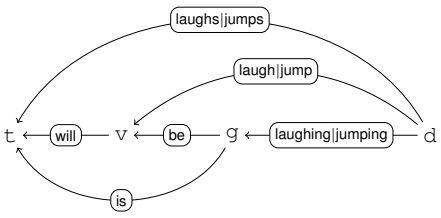
From this data set, the algorithm discussed in section 3 can extract the lexical items shown in (8) by collapsing homophonous items.

(8)

	laughs :: =d.+k.t
Mary :: d.−k	*laughing* :: =d.g
is :: =g.+k.t	*laugh* :: =d.v
will :: =v.+k.t	*jumps* :: =d.+k.t
be :: =g.v	*jumping* :: =d.g
	jump :: =d.v

Merge dependencies in this lexicon can be conveniently visualized as a directed graph. In (9) vertices are category features; each edge corresponds to a lexical item and connects the category of its complement (first phrase it selects) to that of its own.

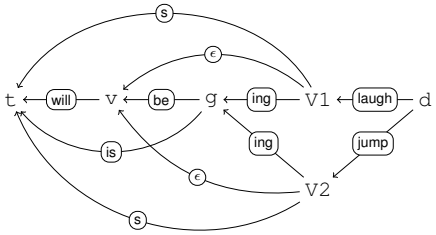
(9)



We begin by decomposing lexical verbs, producing the lexicon in (10). The three lexical items *laughs*, *laughing*, and *laugh* are a valid target for decomposition; and so are *jumps*, *jumping*, and *jump*. Both transitions are motivated both phonologically (factoring out a common prefix) and syntactically (splitting three feature bundles starting with =d).

(10)

Mary :: d.−k	*s* :: =>V1.+k.t
is :: =g.+k.t	*s* :: =>V2.+k.t
will :: =v.+k.t	*ing* :: =>V1.g
be :: =g.v	*ing* :: =>V2.g
laugh :: =d.V1	ϵ :: =>V1.v
jump :: =d.V2	ϵ :: =>V2.v

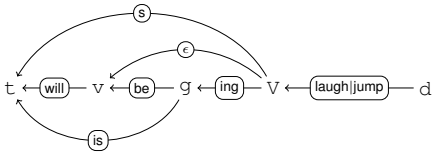


laughing = *laugh* ⊕ *ing* *laughs* = *laugh* ⊕ *s*
jumping = *jump* ⊕ *ing* *jumps* = *jump* ⊕ *s*

This move created two copies each of *s*, *ing*, and ϵ. All of them can be conflated by unifying a single pair of features, V1 and V2, producing a much smaller grammar (11).

(11)

Mary :: d.−k	*laugh* :: =d.V
is :: =g.+k.t	*jump* :: =d.V
will :: =v.+k.t	*s* :: =V.+k.t
be :: =g.v	*ing* :: =V.g
	ϵ :: =V.v

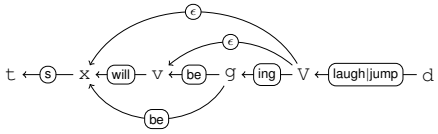


laughing = *laugh* ⊕ *ing* *laughs* = *laugh* ⊕ *s*
jumping = *jump* ⊕ *ing* *jumps* = *jump* ⊕ *s*

The next step targets another repeated sequence of syntactic features: +d.t. This essentially creates a dedicated Tense projection, which hosts the surface position of the subject (12). At this point, concatenation is no longer sufficient for the morphological rules, highlighting the need for a richer theory of morphology.

(12)

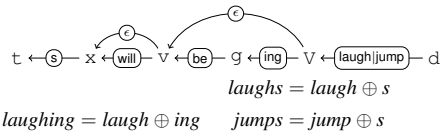
Mary :: d.−k	*laugh* :: =d.V
s :: =x.+k.t	*jump* :: =d.V
be :: =g.x	ϵ :: =V.x
will :: =v.x	*ing* :: =V.g
be :: =g.v	ϵ :: =V.v



$$\begin{aligned} \textit{laughing} &= \textit{laugh} \oplus \textit{ing} & \textit{laughs} &= \textit{laugh} \oplus \textit{s} \\ \textit{jumping} &= \textit{jump} \oplus \textit{ing} & \textit{jumps} &= \textit{jump} \oplus \textit{s} \\ & & \textit{is} &= \textit{be} \oplus \textit{s} \\ & & \textit{will} &= \textit{will} \oplus \textit{s} \end{aligned}$$

This grammar still contains two copies of *be*. While they could be collapsed by unifying `v` and `x`, this move would cause the grammar to overgenerate, producing, for example, the set of ungrammatical sentences *Mary* $(\textit{will})^+$ *be laughing*. However, adding an edge (empty head) from `v` to `x` would make two of these items redundant without generating any unwanted sentences (13). This move can be thought of as decomposing *be* `::` `=g.x` into *be* `::` `=>g.z` and ϵ `::` `=z.x`, where `z` is a fresh feature, and then unifying `z` with `v`. The same is applicable to ϵ `::` `=V.x` and ϵ `::` `=V.v`.

(13)
$$\begin{aligned} \textit{Mary} &:: \texttt{d.-k} & \textit{laugh} &:: \texttt{=d.V} \\ \textit{s} &:: \texttt{=x.+k.t} & \textit{jump} &:: \texttt{=d.V} \\ \epsilon &:: \texttt{=v.x} & \textit{ing} &:: \texttt{=V.g} \\ \textit{will} &:: \texttt{=v.x} & \epsilon &:: \texttt{=V.v} \\ \textit{be} &:: \texttt{=g.v} \end{aligned}$$



$$\begin{aligned} \textit{laughing} &= \textit{laugh} \oplus \textit{ing} & \textit{laughs} &= \textit{laugh} \oplus \textit{s} \\ \textit{jumping} &= \textit{jump} \oplus \textit{ing} & \textit{jumps} &= \textit{jump} \oplus \textit{s} \\ & & \textit{is} &= \textit{be} \oplus \textit{s} \\ & & \textit{will} &= \textit{will} \oplus \textit{s} \end{aligned}$$

We have shown how a Minimalist grammar can be compressed in a way compatible with linguistic theory through repeated application of lexical item decomposition and feature unification. Together they offer a principled way to identify repeating patterns in the lexicon, instantiate them as new lexical items, and collapse any emerging duplicates. Our current work in progress involves building a learning algorithm for syntax with these two operations at its core. This approach would allow to derive (potentially empty) functional heads, producing linguistically motivated generalizations.

References

Noam Chomsky. 1995. *The Minimalist Program*. MIT Press.

Alexander Clark. 2017. Computational learning of syntax. *Annual Review of Linguistics*, 3:107–123.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Heidi Harley and Hyun Kyoung Jung. 2015. In support of the p_{HAVE} analysis of the double object construction. *Linguistic inquiry*, 46(4):703–730.

Yu Hu, Irina Matveeva, John Goldsmith, and Colin Sprague. 2005. Using morphology and syntax together in unsupervised learning. In *Proceedings of the Workshop on Psychocomputational Models of Human Language Acquisition*, pages 20–27. Association for Computational Linguistics.

Makoto Kanazawa. 1995. *Learnable Classes of Categorial Grammars*. Ph.D. thesis, Stanford University.

Masahiro Kawakami. 2018. Double object constructions: Against the small clause analysis. *Journal of Humanities and Social Sciences*, 45:209–226.

Richard S. Kayne. 1984. *Connectedness and Binary Branching*. Foris, Dordrecht.

Gregory M. Kobele. 2018. Lexical decomposition. *Computational Syntax* lecture notes.

Gregory M. Kobele, Travis Collier, Charles Taylor, and Edward P. Stabler. 2002. Learning mirror theory. In *Proceedings of TAG+ 6*, pages 66–73.

Richard K Larson. 1988. On the double object construction. *Linguistic inquiry*, 19(3):335–391.

Jackson Lee and John Goldsmith. 2014. Automatic morphological alignment and clustering. Technical report, Technical report TR-2014-07, Department of Computer Science, University of Chicago.

Jens Michaelis. 2001. *On formal properties of minimalist grammars*. Ph.D. thesis, U of Potsdam.

Craig G Nevill-Manning, Ian H Witten, and David L Maulsby. 1994. Compression by induction of hierarchical grammars. In *Proceedings of DCC'94*, pages 244–253.

David Michael Pesetsky. 1996. *Zero syntax: Experiencers and cascades*. MIT press.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

Jeffrey Mark Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1-2):39–91.

Edward P. Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *Selected Papers from LACL '96*, pages 68–95. Springer Berlin Heidelberg.

Ryo Yoshinaka. 2011. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412(19):1821–1831.